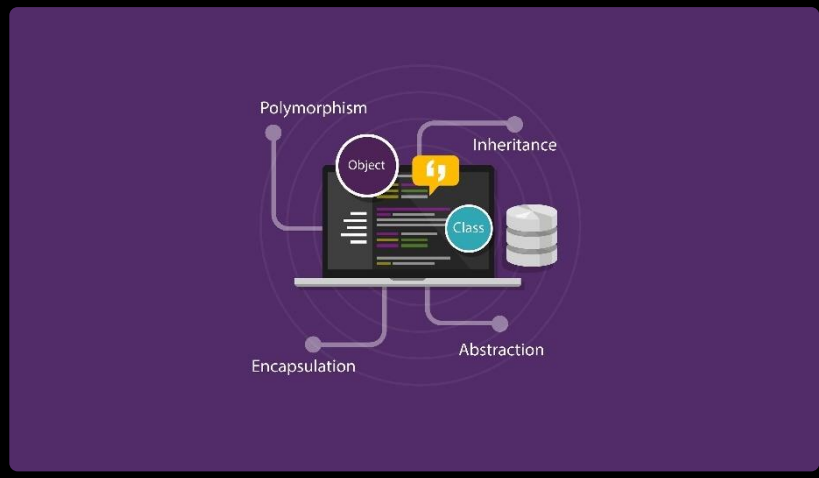
OOP Advanced

Course Overview

SoftUni Foundation



SoftUni Team

SoftUni Team

Software University

http://softuni.bg

Table of Contents

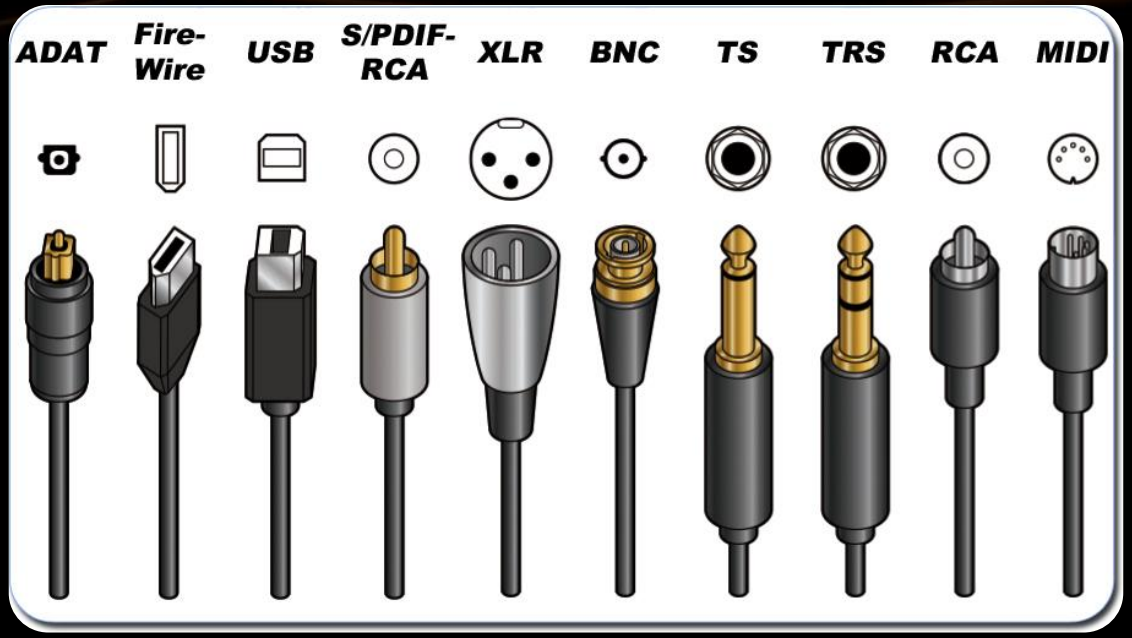
Have a Question?

SoftUni Foundation

sli.do

#JavaFundamentals

SoftUni Foundation

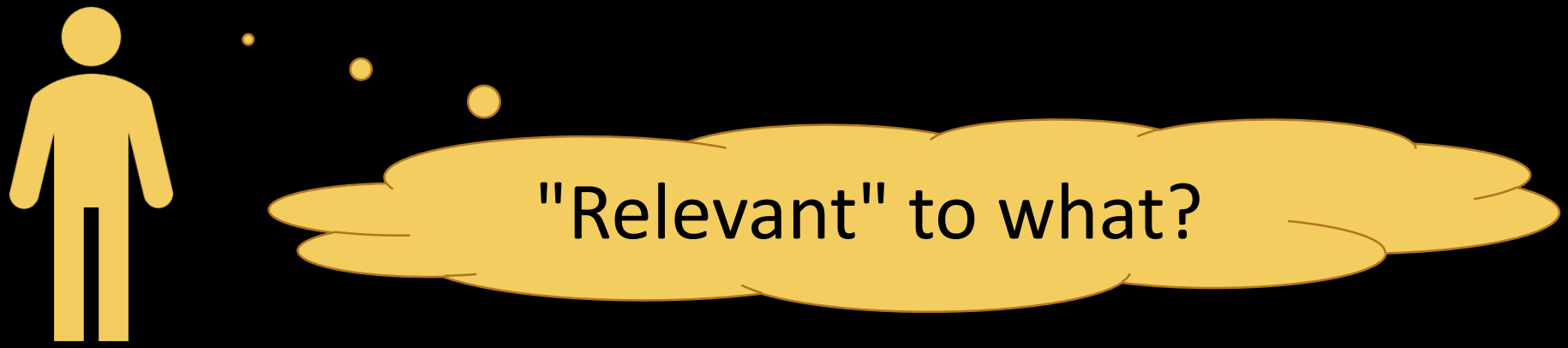


Abstraction

Interfaces

Abstraction in OOP

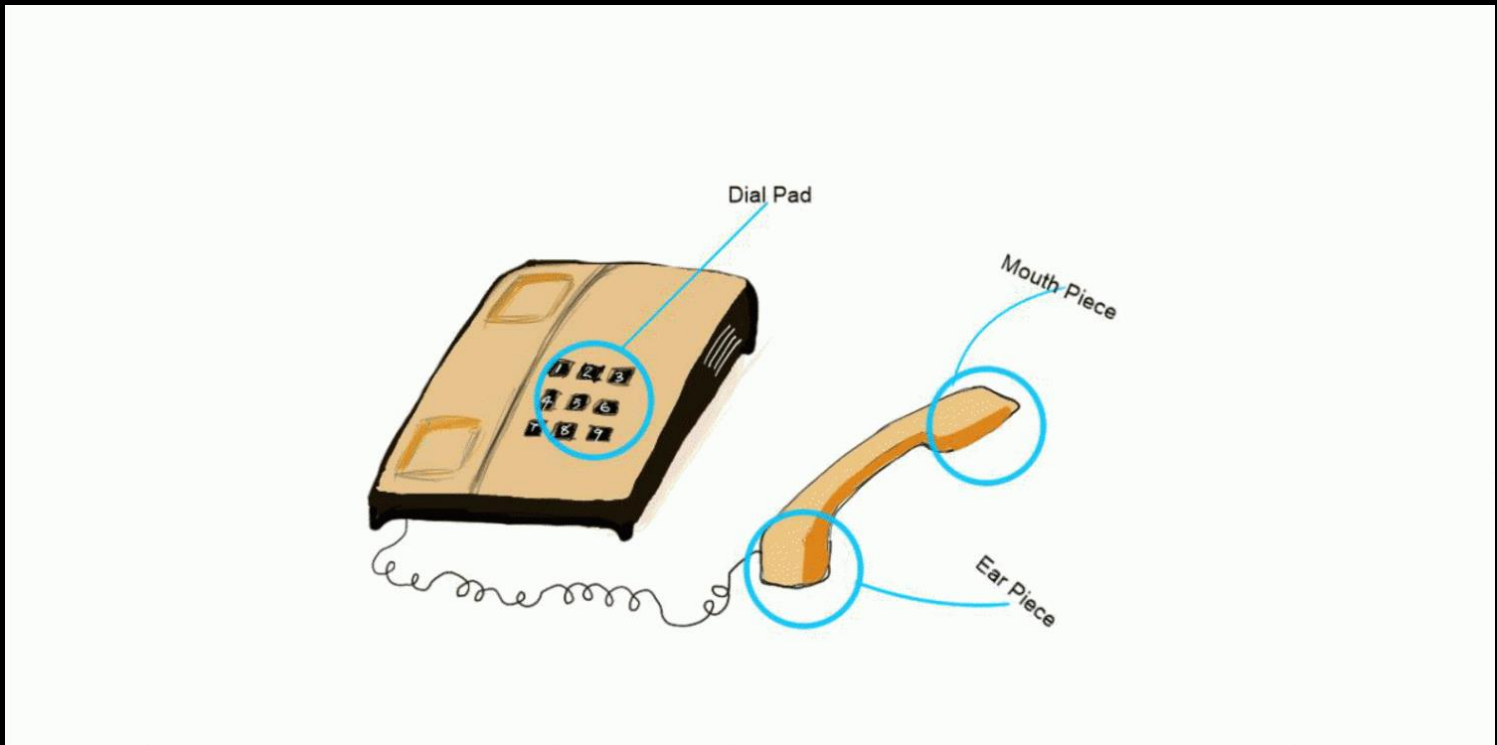
- **Abstraction** means ignoring **irrelevant** features, properties, or functions and emphasizing the **relevant ones**



- **... relevant to the project** we develop
- Abstraction helps managing complexity

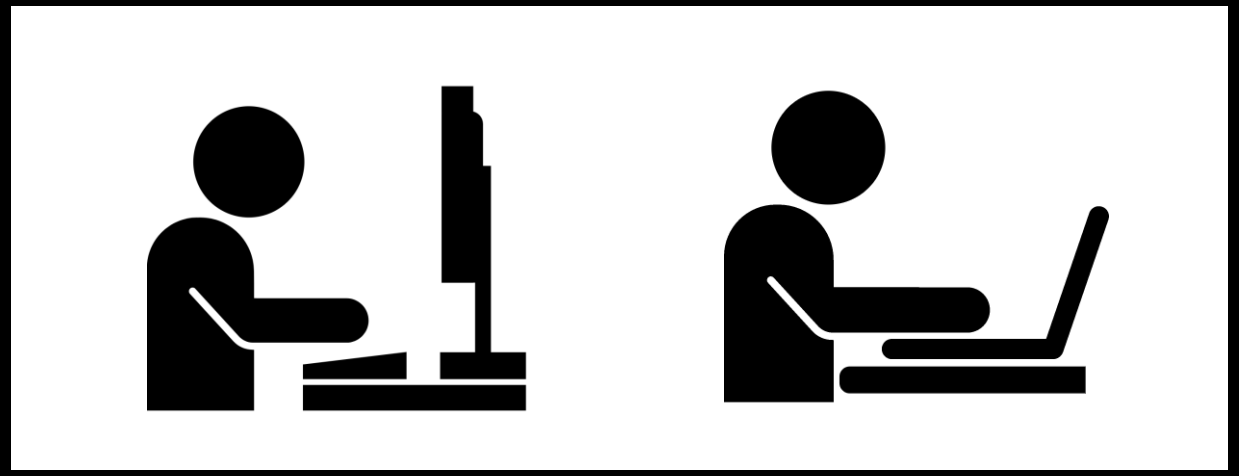
Abstraction Example

- **Abstraction** lets you focus on what the **object does** instead of **how it does it**.



Interface

- Shared boundary across which two or more separate components **exchange information**



SoftUni Foundation

Generics

Reusing Classes

Generics Classes

SoftUni Foundation

- T - used anywhere inside the class

```
public class List<T> {
    public void add(T item) {}
    public T get(int index) {}
}
```

- Your class is **generic** – e.g. you can use it with **any type**

```
List<Apple> apples = new List<>();
apples.add(new Apple);
Apple apple = aplles.get(0);
```

Enumeration

- A set of **predefined constants**

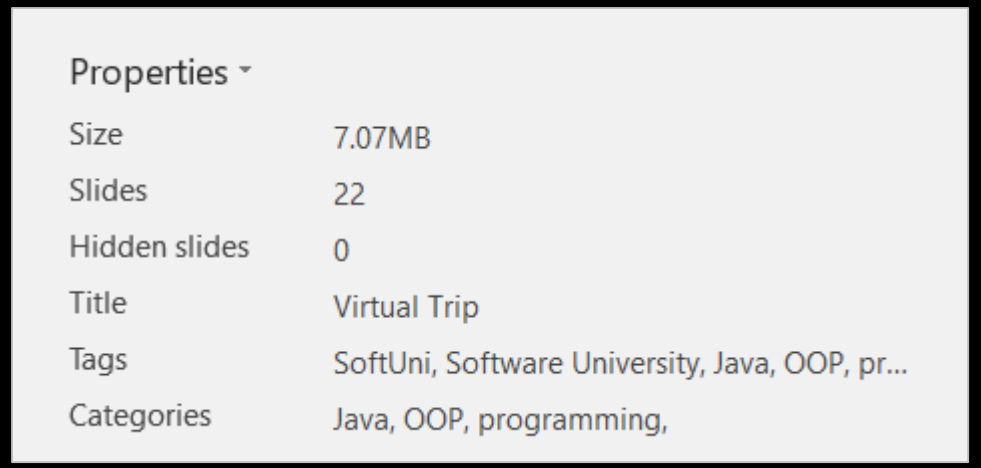
```
public enum Priority {
   HIGH, MEDIUM, LOW
}
```

- Results in a **cleaner code**

```
Error error = new Error(Priority.HIGH);
…
if (error.getPriority.equals(Priority.HIGH)) { … }
```

Annotations

- Provide metadata (data about the data)




```
@Override
String toString() { ... }
```

```
@Test
void testSetAge() { ... }
```

SoftUni Foundation

Reflection

Runtime Type Information

Reflection

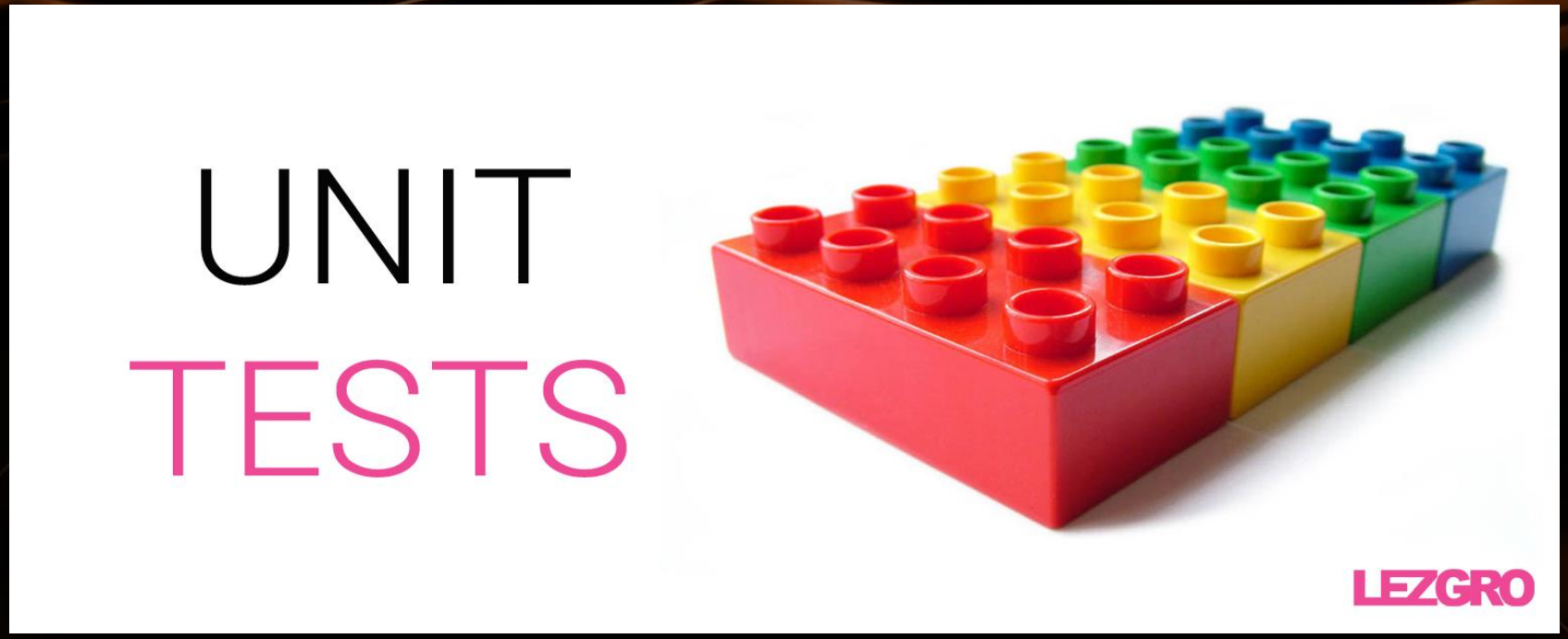
- Can inspect Java classes at **runtime**

```
Class catClass = cat.class
```

- Using the Class

```
int modifiers = catClass.getModifiers();
Class[] interfaces = catClass.getInterfaces();
Method[] method = catClass.getMethods();
Field[] method = catClass.getFields();
Annotation[] annotations = catClass.getAnnotations();
```

SoftUni Foundation



Unit Testing

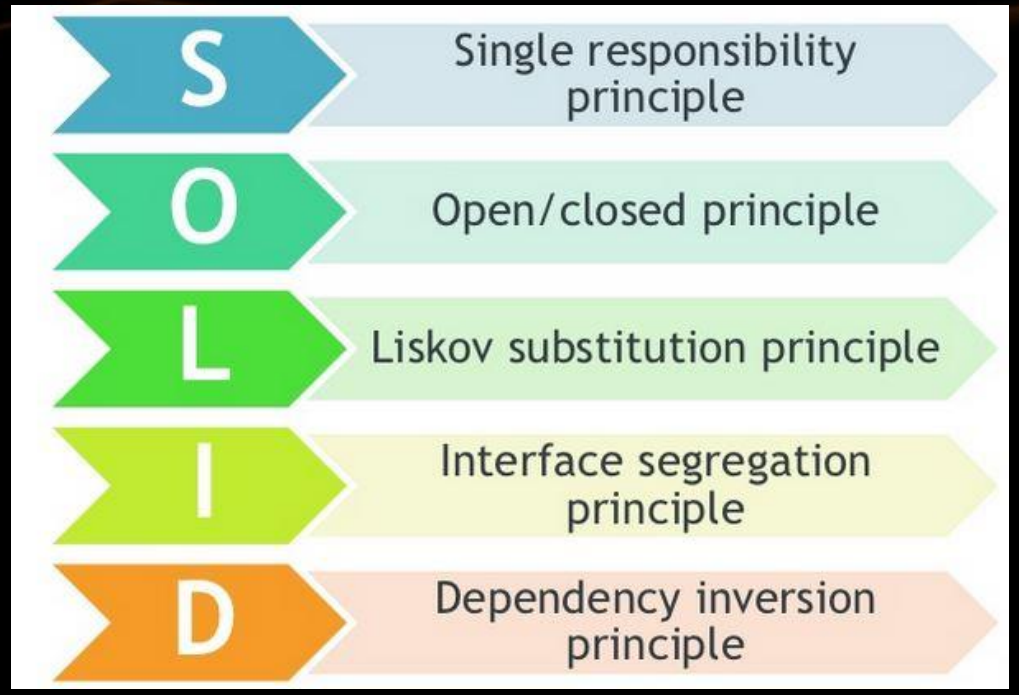
Developer Job

Unit Testing

SoftUni Foundation

- **Individual modules** are tested to find **bugs**
- The main aim is to **isolate each unit** of the system to **identify, analyze and fix the defects**

```
@Test
public void testFeedShouldNotLeaveCatHungry() {
  Cat cat = new Cat();
  cat.feed();
  Assert.assertTrue(cat.isFed());
}
```



SOLID Principles

SOLID

SoftUni Foundation

- **Single responsibility principle**
  - Class should have only a single responsibility
- **Open/closed principle**
  - "software entities … should be open for extension, but closed for modification."
- **Liskov Substitution Principle**
  - Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

SOLID

- **Interface segregation principle**
  - Many client-specific interfaces are better than one general-purpose interface
- **Dependency inversion principle**
  - "depend upon abstractions, NOT concretions"

Course Overview

SoftUni Foundation

Questions?

https://softuni.bg/courses/java-oop-advanced

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

SoftUni Foundation

FORUM